Detection of Anomalies from User Profiles Generated from System Logs

Malcolm Corney **George Mohay** **Andrew Clark**
Information Security Institute
Queensland University of Technology
PO Box 2434, Brisbane QLD 4001, Australia
m.corney@qut.edu.au, g.mohay@qut.edu.au, a.clark@qut.edu.au

Abstract

We describe research into the identification of anomalous events and event patterns as manifested in computer system logs. Prototype software has been developed with a capability that identifies anomalous events based on usage patterns or user profiles, and alerts administrators when such events are identified. To reduce the number of false positive alerts we have investigated the use of different user profile training techniques and introduce the use of abstractions to group together applications which are related. Our results suggest that the number of false alerts that are generated is significantly reduced when a growing time window is used for user profile training and when abstraction into groups of applications is used.

Keywords: User profiling, insider misuse, abstraction.

1 Introduction

Computer crime continues to be problematic for both public and private sectors not only in Australia but at an international level. Over 50% of respondents to the 2006 Computer Security Institute/FBI Computer Crime and Security Survey (Gordon, Loeb et al. 2006) reported unauthorized use of computer systems. In an equivalent Australian survey, the 2006 Australian Computer Crime and Security Survey (AusCERT 2006), 22% of respondents reported experiencing one or more electronic attacks.

The field of computer forensics has been rapidly expanding in the past twenty years in an effort to combat the continuing increase in the incidence of criminal activity involving computers. This field is normally defined around the identification, securing and analysis of evidence for eventual presentation in a court of law. Few cases result in a criminal prosecution and a broader definition of computer forensics can be made that simply attempts to detect, secure and analyse evidence from computer systems. This may be done by an organization, for instance, in response to a security incident, internal or external.

The surveys highlight that the most common types of criminal activity are the results of virus, worm or Trojan infections. Insider abuse of Internet access, email or computer system resources, however, is the third most common type of misuse in the United States of America (Gordon, Loeb et al. 2006) and the second most common type of misuse in Australia (AusCERT 2006).

Insider misuse can be defined as the performance of activities where computers and networks in an organization are deliberately misused by those who are authorized to use them. Some activities which can be categorized as insider misuse include:

- unauthorized access to information which is an abuse of privileges
- unauthorized use of software or applications for purposes other than carrying out one's duties
- theft or breach of proprietary or confidential information
- theft or unauthorized use of staff or customer's access credentials
- computer facilitated financial fraud

This paper reports on work aimed at detecting anomalous events that may be indicators of insider misuse. More specifically we attempt to detect unauthorized use of software applications by users from within an organization. Our general approach has been to build user profiles from computer security audit logs which record the applications used. In particular, we have used the security audit log from computers running the Windows XP operating system for this work.

While we have based this research on a specific operating system, the approach may be generalised to any operating system or computer system, such as an ERP system, which records user's activities as events.

We have created user profiles from data recorded in the Windows Security log by identifying the processes or applications run by a computer user. The events recorded in the Windows security audit log can be correlated to determine when a process was started and terminated by the user or the system. Data from the correlated events can be stored and queried for post hoc investigation of a user's activities on the computer. User profiles include information on which times of the day or week the various applications were used by an individual and also record the first time an application was used by the user.

Users in an organization may have specific duties which they carry out on a routine basis at various times of the working day or week. A sudden departure from routine may be an indicator that the user is not carrying

out their routine duties. Use of applications outside of regular working hours may also be an indicator of misuse of the computer system.

The first use of an application may be an indicator of a user installing software which is outside of an organization's Standard Operating Environment (SOE) if they have the privileges to do so. It may also simply be the user using an application from the SOE for the first time.

We define an alert for this work as the result of detecting a user instigated event which is atypical for that user given their past usage history based on certain criteria. The criteria we have used for generating alerts are based on the situations mentioned above, i.e. first use of an application and usage of an application at times outside the norm for the user.

When the use of new applications or use of applications outside of a user's typical profile is detected alerts can be generated but many of these are likely to be false positives. Any approach which aims to detect anomalous usage must also concentrate on reduction in the number of false positives to have any benefit to an organization.

It is likely that a user's profile will not remain static but will vary as the user's duties change or the organization's SOE changes. It is therefore a requirement of such a system which detects anomalous usage that it be flexible or dynamic in its generation of user profiles. In this paper we present two options for the generation of dynamic user profiles.

The options for creation or training of user profiles and the possibilities for making a user profile dynamic are discussed in Section 2.2.

After user profiles have been generated, each profile can be used as the basis for comparing that user's activities in the period following that on which the profile was created. Any activity out of the ordinary for a profiled user can be flagged to generate an alert to system administrators.

A potential problem in work of this nature is that a large number of alerts may be generated by the simple alert types we defined, and furthermore, many of these may be false positives. In a large organization, large numbers of alerts generated for each and every user may be costly to investigate.

To reduce the number of alerts generated we have created abstractions based on the applications used. The abstractions group similar applications together, rather than using the full and raw process path and name that is recorded in the security audit log.

The first abstraction, which we name *process families*, is created by manually assigning applications with a similar purpose to a family of applications. The second abstraction, process groups, uses a clustering technique to create groups of applications based on the name of the process' executable file and path. Further detail on the design of these abstractions is discussed in Section 2.3.

We show that when these abstractions are used as the basic unit for the user profile rather than the individual processes, the number of alerts generated is reduced. Section 2.3 discusses the development of the abstractions.

Details of our implementation and experimental methodology are given in Section 3. Results for the number of alerts generated from user data have been generated for several profile creation schemes, training periods and levels of event abstraction and are displayed and discussed in Section 4. Section 5 discusses other related work in the field of detection of insider misuse. We draw our conclusions in Section 6 and discuss how this current research may be extended in Section 7.

2 System Design – User Profiling and Event Abstraction

Our aim has been to develop prototype software that implements a capability to identify events that are anomalous and may be indicative of computer misuse within an organization. We have in addition collected data for evaluating the effectiveness and performance of the software and have used it to do so. We worked with data from the Windows security audit log from computers running the Windows XP operating system. When various audit controls are enabled, these logs record information about user log on sessions and the applications or processes invoked by the users of the computers and by the computer system itself.

2.1 Summary of the Design

Our approach consists of six main stages which are summarized below. Further detail of these steps is provided in Section 3 of this paper.

- Data Collection and Preparation – Windows Security log data is collected using a VB script run daily to collect, compress and clear the log file. Further conversion steps are applied to the logs before any data reduction steps are undertaken as the Windows Security log is stored in a proprietary binary format.
- Data Reduction and Correlation – As the amount of data collected from the log sources is voluminous, only events which are recorded as a direct result of user action, such as logging in, logging out and starting and stopping applications, are used for further investigations. The approach taken for correlation is similar to that used by Abbott, Bell et al (2006) where event abstraction is used to recognise logical events from the raw events recorded in the logs.
- Data Storage – For further processing, including preparation of user profiles and alert generation, the data is persistently stored in a relational database.
- Profiling – User profiles are generated so that users' normal or habitual use of applications can be determined.
- Alert generation – Simple alert types are defined based on the data recorded in the user profiles, and usage data for users from ensuing time periods are used for comparison. When abnormal events are detected, alerts are generated.
- Alert checking – When alerts are generated it is necessary for them to be checked to determine if they are benign or if there is some real threat behind the cause of the event.

2.2 Profile Generation and Training

Before a profile can be generated for a particular user we must have that person's usage data for a specific continuous period of time. The training period should be selected so that most of the routine activities a user

performs are included. This will likely be different for different users. For this paper we determine this time period empirically.

We suggest three possible approaches for generating the user profile. The first approach is to use a static or constant window user profile. With this approach detection of alerts is carried out on a weekly basis after the training period and the user profile remains the same for each week of testing. Alerts are always generated based on that initial user profile. This approach is likely to generate many alerts as the person's usage changes due to their role changing within their organization or when software updates are applied. We concede that this is unlikely to be a successful approach but it provides a baseline for comparison with other approaches.

A second approach is to use a growing window, where the profile training time period is continually extended by adding events to the user profile from the testing period after alerts have been generated for the week under test. It is necessary for feedback to be given about whether or not the event causing an alert is benign before it can be added to the training data. With this approach, the user profile becomes dynamic and captures changes in a user's behaviour or in the user's environment due to software updates. A possible problem with this approach is that the user profile may retain too much stale history especially if a user's role changes over time.

A third approach is to use a sliding time window for the user profile where the width of the time window remains constant. After training the user profile and generating alerts based on the data from the testing week, the user profile is recalculated by removing the oldest week of profile data and adding the new week of data that has had any events causing false alerts. This approach is dynamic in nature and does not allow the user profile to become cluttered with too much historical data.

2.3 Event Abstraction

The previous section discussed how we created user profiles. The creation of the profiles is based on the usage of specific processes, that is, specific versions of applications, each of which has its own path and executable name. A limitation with this approach is the high number of alerts that may be generated for a user. A user's activities are likely to change over time as that person's duties or roles change within their organization and the software tools and applications which a person uses is not a stable set. In addition, applications in a user's profile will change as the applications in use are upgraded or changed by the organization.

Some questions which need to be considered then include:

1) Are executables and applications in an organization's SOE similar enough in nature to be grouped together in some way?

2) Should different versions of the same application be grouped together in a user's profile?

For example, a person in a clerical role in an organization would be likely to use tools from an office suite like Microsoft Office, including Word, Excel, PowerPoint and Outlook, and a software developer might use an Integrated Development Environment like Eclipse, and may require the Java Standard Development Kit, an SQL Database server, and a repository tool like Subversion. Each of these applications is likely to be upgraded to the latest versions by the organization as they become available. Alternatively, the organization may change its software procurement policies, meaning that completely different suites of applications may be used.

We propose two types of groupings or abstractions for applications, which we have termed process families and process groups. We compare the use of process families and process groups with individual processes for profile and alert generation.

The first grouping type, *process families*, is based on families of applications or processes that are used for similar purposes, as discussed above, e.g. software development or office administration. All applications and by extension, the process names and their paths that are recorded in the security audit logs, can be assigned to a process family. Further details on how this was implemented are given in Section 4.

The second type of grouping, we have named *process groups*. Process groups are constructed from clusters of applications or processes which are grouped on their path and their executable name. Using this approach processes with the same value for their path or processes with the same value for their executable name will be grouped together. This allows all applications that are run from the same directory along with all version updates to belong to the same process group.

2.4 Alert Checking

Alerts that are generated in a user profiling system would most likely require human processing, although if an organization's security policy specified a Standard Operating Environment, alerts could first be automatically checked against that.

It is desirable in any system that is checking employee's activities that there be as few false alerts as possible, so that system administrators are not wasting time or becoming complacent because they are checking numerous alerts.

In organizations which have a SOE there will be some users who are granted higher level privileges, including the right to install software which is not included in that SOE. In these situations, the user profile is important for reducing the number of alerts that may be generated. If a user profile is recorded for a person, software which is not part of the SOE would only have to be checked the first time an alert is generated. It could be marked as normal or acceptable use and become part of that person's user profile. Once it is part of the user profile, further usage of that software would not cause any alerts to be generated. If the organization did not perform user profiling, and relied on checking all application usage against the SOE, every usage of software not in the SOE would cause an alert to be generated.

3 Implementation and Experimental Methodology

The following sections describe how we implemented the system and the experiments we undertook to find the best set of parameters.

3.1 Data Collection and Preparation

The Windows Security logs collected and examined during the course of the project were from desktop computers running Windows XP Professional Edition with Service Pack 3 installed. Local event logging was enabled and all available auditing options were set. Further to this, auditing of file accesses was enabled where possible at the root level of the logical disks on the computers being logged.

Under normal usage during office hours, with all auditing options enabled, over 1 million events were generated daily. Even though the Security log is meant to be able to be configured to grow in size to 4.5 GB, the maximum log size achieved on the systems under study was approximately 10% of this at 450 MB. The auditing of Object Access with all accesses to all files on the workstation being audited lead to the rapid growth of log files and was turned off after one month of data collection to conserve processing time and storage space.

A VB script was prepared which saved and emptied the log and compressed it to preserve local disk storage. ELDump (Lauritsen 1998) was used to convert the binary event file to a text version for further processing and analysis. Data was collected for a period of nine consecutive months on one of the Windows computers and eighteen months on the other. Table 1 displays the number of login sessions and the number of applications started for each user on the computer systems being surveyed.

Computer Name	Days of Data Collection	Number of Login Sessions	Number of Processes Started
GANDALF	297	348	27,821
ARAGORN	531	124	953,429

Table 1: Information about logged data for the computers studied

3.2 Data Reduction and Correlation

A great deal of information useful for extraction of a user's activities is recorded in the Windows Security log. All events in the Windows Security log contain some common data, such as the date and time stamp, computer name, domain name, user name, event type and identifier numbers, and further information specific to each type of event.

For user profiling, the most useful event types include log on, log off, process start and process exited events. These provide details of a user's log-in sessions and interactions with the applications and services installed on the computer.

It is necessary to correlate log on and log off events to determine the duration of a user's log in session. This is possible in the Windows Security log by matching the value in the *Session Identifier* field of the event. Similarly, the process start and process exited events from a specific user and log in session can be matched to determine the duration of a user's application usage. This can be done by matching the Session Identifier, Process Identifier and Process Name for the relevant events. When the matching process exited event occurs, the duration for which the process ran can be recorded.

When full auditing is enabled in the Windows Security log, Object (file handles, network resources etc.) accesses are recorded in the log and these are recorded with the specific process name and process identifier which accessed the object. Many hundreds of object access events are recorded while an application is being used. For this work, the object access events were neglected due to the complexity of data recorded in them.

3.3 Data Storage

Relational database tables were used to store information relating to users and their activities on the computers studied. Information from the Windows Security logs was stored for each user of each computer. This entailed an entry for each log-in session including the start and end times of the session and the total number of applications invoked by the user during that session. Entries were also stored for each application including the name and full path of the application, start and end times of the application, and details of the parent process responsible for invoking the application. In many cases, the parent process is `Windows\explorer.exe` as this is the desktop application that the user interacts with when using the Windows XP operating system. For example, when the Eclipse IDE application is started from the desktop, `Windows\explorer.exe` is recorded as the parent process of the `Eclipse\eclipse.exe` process. Many processes, however, are spawned by an application a user may have started. For example, when a user starts a Java application from within the Eclipse IDE, the `Java\bin\javaw.exe` process is invoked to run the Java application but it is invoked by the `Eclipse\eclipse.exe` process. This information could allow complex hierarchies of process usage to be determined. The hierarchical nature of the processes was not analysed for inclusion in this paper.

3.4 User Profiles

To generate user profiles we collected data on the usage of which applications a person started. In particular, we chose to record the following attributes which were used for the generation of alerts:

1. The hour of the day an application was started;
2. The day of the week an application was started; and
3. Whether or not the application had been run by the user previously

A time period for user profile construction also had to be considered. Figure 1 shows the cumulative number of new processes used by different users for the data we collected. There are a large number of processes recorded for the first time, for a user in the first eight weeks or so of the logs and then the rate of new processes recorded tapers off slightly.

We tested each of the user profile types introduced in Section 2.2, namely constant time window, growing time window and sliding time window. For these experiments, we tested each combination of profile type and process

abstraction type with 14, 28 and 56 day initial time periods for the user profile.

To ensure that all experiments could be compared with each other, the end date for the training period of the initial time window was set as a constant. All alerts generated in successive weeks were based on the same set of usage data. This means that the starting date of the initial time window for the 56 day training period precedes the starting date of the 14 day training period by 42 days.

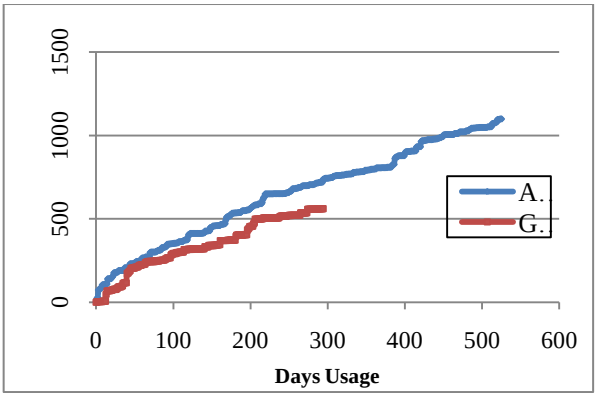

Figure 1: Cumulative Count of New Processes Used by a User

3.5 Alert Generation

Once the different profiles were created, alerts were calculated for a seven day testing period directly following the profile training period. The total number of alerts for the week for each of the attributes measured, i.e. hour of day that a process was started, day of week that a process was started, and if a process was started for the first time, were recorded.

For the purposes of comparison of different approaches in our system, all events causing alerts have been considered to be benign or in other words, false positives. After the testing was carried out on each week's data, the dynamic user profile types were regenerated to include all events from the week's data that was just tested. The results reported in Section 4 for each experiment are the cumulative number of alerts for a period of thirty weeks of alerting for each of the two users for which we had data.

3.6 Abstraction

As discussed in Section 2.3, we have proposed two abstractions for collections of processes for reduction of the number of alerts generated by our system. Our experiments have also recorded the results when the abstractions were not used.

The first of the abstractions we proposed was process families. For the security audit logs collected from the computers under study, nine process families were created from 2,204 differently named processes. The nine process families were: operating system tools, office applications, games, security tools, browsers, development tools, servers, utility applications and installers. This was a laborious and manual task, although some of the processing could have been automated, e.g. all applications in a particular directory hive could have been labelled as part of the same family. In a large organization, such an approach without automation would be prohibitively expensive to deploy.

The second abstraction proposed was process groups. These groups were generated programmatically using a clustering approach. For each distinct process name recorded in the security audit logs, the full process path was extracted and split into a path name and a process name. For example, the process `C:\Program Files\Java\jdk1.6.0_17\ bin\java.exe` has a path value of `C:\Program Files\Java\jdk1.6.0_17\bin` and a process name value of `java.exe`. All applications with the same path as the `java.exe` executable are considered part of the process group, e.g. `javaw.exe` and `javac.exe`. Continuing with the construction of this process group, applications from different versions of the Java SDK `bin` directory are grouped together:

- `C:\Program Files\Java\jdk1.6.0_17\bin\java.exe`
- `C:\Program Files\Java\jdk1.6.0_17\bin\javaw.exe`
- `C:\Program Files\Java\jdk1.6.0_16\bin\java.exe`
- `C:\Program Files\Java\jdk1.6.0_16\bin\javaw.exe`
- `C:\devel\jdk1.6.0_10\bin\java.exe`
- `C:\devel\jdk1.6.0_10\bin\javaw.exe`
- `...`

The process groupings were constructed for the data sets collected from each computer. The number of process groups formed and the original number of distinct processes are recorded in Table 2. The distinct process names are based on the full path and executable name as recorded in the computer logs. This table also records the number of distinct executable names and the number of distinct directories in which those executables were located.

Computer	GANDALF	ARAGORN
Distinct Processes	561	1099
Distinct Executables	424	747
Distinct Directories	255	576
Process Groups	153	271

Table 2: Number of Process Groups Formed

For both process families and process groups, the alerting process using the three different profile types for user profile generation were tested and compared with the results where no abstraction of processes was attempted for each of the three initial time window period.

4 Experimental Results

The number of alerts for the three alert types (new process, new hour for process and new day for process) were collected for each of the three profile type training schemes (constant window, moving window and growing window) for initial training periods of 14, 28 and 56 days. The alerts were generated for the data sets collected from both computers under study for a period of 30 weeks with regeneration of the dynamic profile types after each week of testing.

These results were collected for processes as the baseline case, and for process families and process groups to determine the effect of the different abstraction mechanisms.

In all tests, the starting date for alert generation for a particular computer began on the same date, so that the effect of training period could be compared. This means that the training periods extended back to different starting dates for the different training periods. The total number of alerts for the thirty week period for each of the tests conducted is displayed in Tables 3, 4 and 5. The results are discussed in further detail in the following paragraphs.

4.1 Baseline Case – All Processes

Table 3 displays the number of alerts generated for the different training approaches for user profile generation for both computers, using three different initial training time windows.

It can be seen for the three different training approaches when longer initial training periods were used, fewer alerts were generated. This is an expected result as the trained profile contains more information when a longer training period is used.

The results indicate that the training approach which generates the fewest alerts is the Growing Window approach. This approach is dynamic in nature and retains all past history for a user. While the Moving Window approach is also dynamic, it would appear from the results that when past history is removed from the user profile, higher numbers of alerts are generated. The Moving Window approach generated fewer alerts than the Constant Window approach for one computer but the results were reversed for the other.

Training Period (days)		GANDALF			ARAGORN		
		New	Day	Hour	New	Day	Hour
14	Constant Window	425	484	466	514	581	581
28		412	477	465	495	567	565
56		301	378	390	454	543	549
14	Moving Window	380	428	424	585	600	601
28		362	417	411	549	582	586
56		346	398	405	525	568	574
14	Growing Window	191	295	284	450	534	536
28		180	283	278	495	564	563
56		165	248	267	454	541	547

Table 3: Number of Alerts Generated over 30 Weeks for Processes

4.2 Process Families

Table 4 shows the results of alert generation when processes were grouped together into process families as outlined in Section 2.3.

It can be seen that the frequency of alerts dropped significantly when processes were aggregated into process families. This level of aggregation is clearly too coarse to be useful. As soon as one process belonging to a process family is used, no other processes from that family will generate an alert.

Training Period (days)		GANDALF			ARAGORN		
		New	Day	Hour	New	Day	Hour
14	Constant Window	2	6	3	2	3	4
28		1	3	2	0	2	3
56		1	2	2	0	1	3
14	Moving Window	3	9	9	3	7	6
28		2	8	4	0	3	3
56		2	4	3	0	1	3
14	Growing Window	2	3	3	2	3	4
28		1	2	2	0	2	3
56		1	2	2	0	0	3

Table 4: Number of Alerts Generated over 30 Weeks for Process Families

4.3 Process Groups

Table 5 displays the results of the number of alerts generated when processes were clustered together into process groups using the approach outlined in Section 2.3. It can be seen from the results that the number of alerts from the 30 week testing period is significantly lower than when individual processes were tested. The number of alerts generated for the growing window profiles with eight weeks of training data produce on average three or four alerts per user per week. For a large organization this could still be a significantly high number of alerts in total but it is a significant improvement on the average of five to twenty alerts per user per week when no aggregation is used.

Training Period (days)		GANDALF			ARAGORN		
		New	Day	Hour	New	Day	Hour
14	Constant Window	96	129	120	126	159	154
28		88	123	118	115	143	139
56		85	112	112	98	134	130
14	Moving Window	122	142	141	155	164	165
28		117	141	133	133	149	153
56		97	120	118	126	145	148
14	Growing Window	112	141	131	98	131	129
28		105	134	123	98	128	127
56		85	109	112	89	119	115

Table 5: Number of Alerts Generated over 30 Weeks for Process Groups

The results for process groups also indicate that the best technique for user profile generation as measured by reducing the number of alerts is the Growing Window. This is discussed further in Section 6.

5 Related Work

There are quite comprehensive event monitoring and event correlation products on the market. However they are typically platform specific and focus generally on network event correlation and/or centralized event monitoring and log management rather than *post hoc* correlation of events for forensic purposes.

We note also a considerable body of research in the area of security event correlation, ranging from alert correlation in intrusion detection systems (Ning, Cui et al. 2004; Morin, Mé et al. 2009) through to the standardization and formatting of audit or log records (Bishop 1995; Kent and Souppaya 2006) and audit reduction (Pfleeger and Pfleeger 2003).

Specific related work in insider misuse detection quite commonly has implemented systems aimed at specific operating systems without mention of the system's applicability for other operating systems (Christoph, Jackson et al. 1995), or have developed approaches not aimed at operating systems at all, e.g. for database systems (Chung, Gertz et al. 1999). It is also quite common for researchers to use simulated data (Maybury 2006; Anderson, Selby et al. 2007).

Security personnel at the Los Alamos National Laboratory (Christoph, Jackson et al. 1995) implemented an approach to detect security policy violations on computer systems. This was capable of detecting activities by insiders abusing operating system privileges and outsiders attempting to gain clandestine access. They produced Network Anomaly Detection and Intrusion Reporter (NADIR) and UNICOS Real-time NADIR (UNICOS) to summarize user and system activity profiles. This system was aimed specifically at the UNICOS operating system and no extensibility was considered.

Chung, Gertz et al (1999) created a misuse detection system for relational databases. They computed user profiles from audit log data in an attempt to detect insider misuse of a financial database system in use in a bank. They present scenarios based on a bank teller misusing their privileges to gain customer credit card information or to transfer customer funds to their own account.

Another work by Shavlik, Shavlik et al. (2001) focused on profiling and identifying Windows 2000 users via keystroke dynamics. This work was intended to complement insider misuse detection rather than to detect insider misuse.

Maybury (2006) reported on a collaborative, six month workshop to characterize and create analysis methods to counter sophisticated malicious insiders in the United States Intelligence Community. His paper discusses a generic model of malicious insider behaviours, distinguishing motives, (cyber and physical) actions, and associated observables. The paper outlines several prototype techniques developed to provide early warning of insider activity, including novel algorithms for structured analysis and data fusion and reports performance assessment in an operational network against simulated insiders (an analyst, application administrator, and system administrator).

Anderson, Selby et al (2007) report on their behaviour profiling and misuse detection system, IRIS, which involves an intricate architecture of components to achieve real-time anomaly detection based upon a variety of inputs including operating system logs. IRIS employs the proprietary MQ Telemetry Transport protocol (MQTT) implemented by the IBM MQ MicroBroker. The system has been deployed to date only with simulated data and the authors note that their system requires a comprehensive set of user data, suggesting periods of time of the order of years.

Cathey, Ma et al (2003) concentrate on the detection of insider misuse in information retrieval systems. Their rules based approach is based on the creation of user profiles and relies on each user's profile recording the types of documents that the user is allowed to retrieve from the information retrieval system. Again, this work is quite different to the research we have undertaken.

Ma and Goharian (2004) built user profiles to detect misuse in search systems based on activities learnt through clustering and relevance feedback. Goharian and Ma (2005) showed that they could achieve equivalent results to Cathey, Ma et al (2003) in detecting off-topic accesses to files in an information retrieval system by using a subset of the features that were originally proposed. The research presented in these approaches has focused on the detection of insider abuse of privileges by detecting anomalous behaviour from access control lists either prescribed by system administrators or from definitions generated from "learnt normal" behaviour. We note that these approaches have not targeted an operating system's Security Audit event logs and the applications used for generation of user profiles but have concentrated on file and document accesses.

Magklaras and Furnell (2006) have produced a threat prediction specification language for modelling insider threat and intrusion incidents. This approach is quite different to the anomaly detection approach which we have presented in this paper.

Other researchers in the field of insider misuse detection have defined the threat of insider misuse (Bishop and Gates 2008; Pfleeger and Stolfo 2009) but have not moved to implementation. There have also been some frameworks for insider misuse proposed (Baek, Kim et al. 2008; Zhang, Ma et al. 2009) but again, these have not been fully implemented.

None of these papers have discussed the likelihood of high levels of false positives or ways to address this problem. There has been no mention in any of these papers of the different approaches we have presented for training user profiles and there has been no discussion of abstractions for collections of user applications.

6 Conclusions

We have shown that it is possible to build user profiles from data recorded in the Security Audit logs of computers running the Windows XP Operating System. The main information used from the events logged has been the names of the applications invoked by the users of the computers. It should be possible to use the same approach for any computers whose operating system records similar data.

We outlined three different approaches to generating user profiles, where a constant window or a growing window or a moving window can be used to specify the training time period used in the user profile. A user's profile may change due to a change in their role or may be due to software upgrades or changes imposed by the user's organization. Ideally such changes should not cause a major increase in the number of alerts generated by the system.

Our experimental results indicate that the growing window approach generates fewer alerts than the moving window approach and its dynamic nature captures changes in usage of applications without generating an excess number of alerts. This is thought to be due to the growing window approach not losing any past history from the user profile.

Our results also indicate that a longer training period will result in a richer user profile which has the effect of generating fewer alerts.

We also proposed and implemented two groupings of processes in an effort to reduce the number of alerts generated by the system. One of these, Process Families where all applications used on the computers were placed in a small number of groups, was too coarse in its groupings to be useful.

The second approach we named Process Groups and created by clustering together applications with the same path and/or the same executable name. This approach resulted in a significant reduction in the number of alerts when compared to individual process names, three to four alerts per user per week. We conclude that the use of this abstraction provides a positive benefit to the overall system

It is of course necessary to investigate any alerts that are generated by a system such as the one described in this paper. If an organization employs a Standard Operating Environment, alerts generated from the first invocation of a process or process group could be automatically checked against the list of software in the SOE. Such a check should result in a further reduction in the number of alerts that need to be manually checked.

All user profiles have been generated for individual users. Using the abstraction approach it should be possible to make comparisons between users based on the collection of processes in their profile or alternatively to create profiles for an organization. Based on the role of similar people in the organization, a profile could be assigned based on the expected usage of applications. The profile could then be dynamically updated as the user settles in to their work routine. If this were done, there would be no need to wait for a training period for a new employee in an organization before their usage is routinely monitored. Care would have to be taken with such an approach that the user profile is properly regenerated once the employee has been working for the training window time period.

We believe that the approach outlined here for users of PCs could also be applied to other sources of data. Any source of data where user's actions are recorded could be used, e.g. web proxy logs where people's web browsing habits are recorded or ERP systems where user transactions are recorded. A possible application for this approach with ERP systems is the detection of financial fraud.

7 Future Work

In future work we intend to improve the system by changing and improving the way process groups are formed. When the process group clusters were formed, it was noticed that some of the clusters contained a single process name. A further reduction in alerts may be achieved by aggregating single processes by using their process family rather than their process group, i.e. by using a mixed model of aggregation.

Alternatively we could use some other means to determine if sufficient parts of a process' path are the same for the process to belong to an existing process group. This could be implemented as a set proportion of the path or it could use an edit distance measure with a set threshold determining whether two processes belong in the same process group.

It should be noted that the work carried out to date is based on the presence or absence of a particular application in a user's profile. In future work we will utilise the frequency of usage of applications, allowing us to build probabilistic models of usage for each person.

In our current work we have recorded but not made use of the amount of time that a user uses particular applications or groups of applications in their routine work. This is calculated from the start and stop times recorded for each process in the Security Audit log. By measuring the amount of time an application is used, better models of typical usage for a user can be built and this will be incorporated in our future work.

We intend to investigate the proportion of time a process is actively used while it is open by making use of a library which allows the amount of CPU time an application uses to be monitored to provide a more suitable measure of application usage. Keystroke logging tools may also provide a solution here as they can record which application is receiving key strokes and mouse clicks. Further investigation into monitoring application usage is therefore warranted.

References

Abbott, J., Bell, J., Clark, A., de Vel, O. and Mohay, G. (2006). Automated recognition of event scenarios for digital forensics. *2006 ACM Symposium on Applied Computing*, Dijon, France.

Anderson, G. F., Selby, D. A. and Ramsey, M. (2007). Insider attack and real-time data mining of user behavior. *IBM Journal of Research and Development* **51**(3/4):465-475.

AusCERT. 2006 Australian computer crime and security survey: http://www.auscert.org.au/images/ACCSS2006.pdf. Accessed 12 Oct 2007.

Baek, E., Kim, Y., Sung, J. and Lee, S. (2008). The design of framework for detecting an insider's leak of confidential information. *e-Forensics*, Adelaide, Australia.

Bishop, M. (1995). A standard audit trail format. *National Information Systems Security Conference*, Baltimore, USA.

Bishop, M. and Gates, C. (2008). Defining the insider threat. *Fourth Annual Cyber Security and Information Intelligence Research Workshop* Oak Ridge, USA.

Cathey, R., Ma, L., Goharian, N. and Grossman, D. (2003). Misuse detection for information retrieval systems. *2003 ACM CIKM International Conference on Information and Knowledge Management*, New Orleans, USA.

Christoph, G. G., Jackson, K. A., Neuman, M. C., Siciliano, C. L. B., Simmonds, D. D., Stallings, C. A. and Thompson, J. L. (1995). UNICORN: Misuse detection for UNICOS. *1995 ACM/IEEE Conference on Supercomputing*, San Diego, USA.

Chung, C. Y., Gertz, M. and Levitt, K. (1999). DEMIDS: A misuse detection system for database systems. *Third Working Conference on Integrity and Internal Control in Information Systems: Strategic Views on the Need for Control*, Amsterdam, The Netherlands.

Goharian, N. and Ma, L. (2005). On off-topic access detection in information systems. *14th ACM International Conference on Information and Knowledge Management*, Bremen, Germany.

Gordon, L. A., Loeb, M. P., Lucyshyn, W. and Richardson, R. 2006 CSI/FBI computer crime and security survey: http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2006.pdf. Accessed 12 Oct 2007.

Kent, K. and Souppaya, M. Guide to computer security log management: http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf. Accessed 6 Jun 2007.

Lauritsen, J. ELDump: http://www.ibt.ku.dk/jesper/ELDump/default.htm. Accessed 1 Apr 2005.

Ma, L. and Goharian, N. (2004). Using relevance feedback to detect misuse for information retrieval systems. *Thirteenth ACM International Conference on Information and Knowledge Management*, Washington, USA.

Magklaras, G. B. and Furnell, S. M. (2006). Towards an insider threat prediction specification language. *Information Management & Computer Security* **14**(4):361-381.

Maybury, M. (2006). Detecting malicious insiders in military networks. *MILCOM-06*, Washington, USA.

Morin, B., Mé, L., Debar, H. and Ducassé, M. (2009). A logic-based model to support alert correlation in intrusion detection. *Information Fusion* **10**(4):285-299.

Ning, P., Cui, Y., Reeves, D. S. and Xu, D. (2004). Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security* **7**(2):273-318.

Pfleeger, C. P. and Pfleeger, S. L. (2003): *Security in computing*. Upper Saddle River, Pearson Education, Inc.

Pfleeger, S. L. and Stolfo, S. J. (2009). Addressing the insider threat. *IEEE Security and Privacy* **7**(6):10-13.

Shavlik, J., Shavlik, M. and Fahland, M. (2001). Evaluating software sensors for actively profiling Windows 2000 computer users. *4th International Symposium on Recent Advances in Intrusion Detection*, Davis, USA.

Zhang, H., Ma, J., Wang, Y. and Pei, Q. (2009). An active defense model and framework of insider threats detection and sense. *Fifth International Conference on Information Assurance and Security*, Xi'an, China.